# 3-D Deformable Grid Electromagnetic PIC for Parallel Computers

J. Wang and P.C. Liewer

Jet Propulsion Laboratory, California Institute of Technology, Pasadena

S. R. Karmesin* and 1). Kondrashov

California Institute of Technology, Pasadena

## 1. Introduction

Plasma particle-in-cell (PIC) codes [Birdsall and Langdon,1985] have become standard research tools in plasma physics research. A PIC code models a plasma as many test particles and follows the evolution of the orbits of individual test particles in the self-consistent electromagnetic fields. Each time step in a PIC code consists of two major stages: the ***particle push*** to update the particle orbits and calculate the new charge and/or current density, and the ***field*** solve to update the electromagnetic fields. Two particle- grid interpolation steps are used to link particle orbits and the field components: a "gather" step is used to interpolate fields from grid points to particle positions and a "scatter" step to deposit the charge/current of each particle to grid points.

Most existing PIC codes are based on the use of orthogonal grid (Cartesian or cylindrical). This restricted their applications to problems with simple geometries. In this paper we describe a new electromagnetic PIC algorithm that uses body-fitted curvilinear coordinates for large-scale simulations of problems involving complex geometries (e.g. high power microwave devices) on parallel supercomputers.

Our numerical formulation is based on the use of deformable hexahedral cells which are logically connected. We define a physical space where the cells can be deformed in shape to body-fit complex geometries and a logical space consisting of a Cartesian mesh Each grid point in the physical space is mapped one-to-one to the Cartesian mesh in the logical space. All physical quantities are computed in the physical space However, particle tracing as well as the gather/scatter are performed in the logical space.

We find that the deformable hexahedral grid is sufficient to handel a wide variety of complex geometries. For instance, shown in Fig.1 is a cylindrical domain constructed using deformed hexahedral cells. There are several advantages of using the deformable hexahedral cells than using more sophisticated grids, such as tetrahedral cells or unstructured grids. The major advantage is that the logically Cartesian connection of the cells preserves the nearest neighbor relationships and avoids the complications of indirect addressing of unstructured grids. The location in memory of quantities defined in neighboring cells can be easily computed without memory references. (This is in contrast with an instructed grid where the neighbors of a given cell must be found by lookups in a table.) Hence, particles can be located almost as efficiently as that in an orthogonal grid PIC code. for gather and scatter calculations. This is especially important for large scale simulations because a PIC code typically spents most of its computing time to push particles. The use of a Cartesian logical space also significantly simplifies the procedures for particle-grid interpolation on non-orthogonal grids. Gather and scatter algorithms developed for Cartesian grid baaed electromagnetic PIC codes (for example, the rigorous charge conservation current deposit) maybe utilized directly. (For tetrahedral and unstructed grids, these interpolations can be very complicated.) Finally, the implementation of the code on to parallel supercomputers is almost identical to that of the Cartesian grid based parallel EMPIC code [Wang et al.,1995].

The numerical algorithm is described in Section 2. In Section 3, we perform numerical tests of the code and study the accuracy of the algorithm The performance of the code on a MIM 1) parallel supercomputer is discussed in Section 4. Section 5 contains a summary and conclusions.

*NOW at K2 Research Inc., Pasadena, CA

## 2. Algorithm

### Governing Equations

'The governing equations in an electromagnetic PIC code are Newton's second law for individual particle trajectories:

$$\begin{aligned} \frac{d\gamma m\mathbf{v}}{dt} &= \mathbf{F} = q(\mathbf{E} + \mathbf{v} \times \frac{\mathbf{B}}{c}) \\ \frac{d\mathbf{x}}{dt} &= \mathbf{v} \end{aligned} \quad (1)$$

where $\gamma m = m/\sqrt{1 - v^2/c^2}$ is the relativistic mass of the plasma particle, and the Maxwell's equations for the macroscopic electromagnetic field:

$$\begin{aligned} \frac{\partial \mathbf{B}}{\partial t} &= -c\nabla \times \mathbf{E} \\ \frac{\partial \mathbf{E}}{\partial t} &= c\nabla \times \mathbf{B} + \mathbf{J} \\ \nabla \cdot \mathbf{B} &= 0 \\ \nabla \cdot \mathbf{E} &= \rho \end{aligned} \quad (2)$$

The charge density $\rho$ and the current density J are derived from the motion of plasma particles and satisfy the charge continuity equation:

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot \mathbf{J} \quad (3)$$

Note if this charge conservation condition can be satisfied rigorously numerically, one can update the electromagnetic field using only the two curl Maxwell's equations.

To obtain a numerical solution, the curl Maxwell equations arc recast. in the integral form by integrating curl equations over a surface S,

$$\begin{aligned} \partial_t \int_s \mathbf{B} \cdot \mathbf{ds} &= -c \oint_l \mathbf{E} \cdot \mathbf{dl} \\ \partial_t \int_s \mathbf{E} \cdot \mathbf{ds} &= c \oint_l \mathbf{B} \cdot \mathbf{dl} + \int_s \mathbf{J} \cdot \mathbf{ds} \end{aligned} \quad (4)$$

and the the divergence equations and the conservation of charge equation over a volume V,

$$\begin{aligned} \oint_s \mathbf{E} \cdot \mathbf{ds} &= \int_v \rho dv \\ \oint_s \mathbf{B} \cdot \mathrm{ds} &= \mathrm{o} \\ \partial_t \int_p dv &= \int \mathbf{J} \cdot \mathbf{ds} \end{aligned} \quad (5)$$

### Computation Grid

The physical space $\Omega_P$ is discretized into a structured grid of deformable hexahedral cells which are logically Cartesian. The cells in the physical space *may* be distorted to fit complex geometries. However, each cell in $\Omega_P$ is mapped one-to-one to a unit cube in a logical space $\Omega_L$. The vertices of the cells are at the locations where the logical coordinates are all integers(See Fig. 2). The neighbor can be found by simply incrementing indices.

Let r = *(r,s, t)* denote the the logical space coordinates and x = $(x, y,$ z) the physical space coordinates. A simple tri-linear interpolation is used to map the logical coordinates to the physical coordinates. For instance, the mapping to the $x$ coordinate inside cell (i, j, *k)* is

$$x = x_0 + x_1 r + x_2 s + x_3 rs + x_4 t + x_5 rt + x_6 st + x_7 rst \quad (6)$$

where

$$\begin{aligned} x_0 &= x_{i,j,k} \\ x_1 &= x_{i+1,j,k} - x_{i,j,k} \\ x_2 &= x_{i,j+1,k} - x_{i,j,k} \\ x4 &= x_{i,j,k+1} - x_{i,j,k} \\ x3 &= x_{i+1,j+1,k} + x_{i,j,k} - x_{i,j+1,k} - x_{i+1,j,k} \\ x_5 &= x_{i+1,j,k+1} + x_{i,j,k} - x_{i,j,k+1} - x_{i+1,j,k} \\ x_6 &= x_{i,j+1,k+1} + x_{i,j,k} - x_{i,j,k+1} - x_{i,j+1,k} \\ x_7 &= -x_{i,j,k} + x_{i+1,j,k} + x_{i,j+1,k} + x_{i,j,k+1} \\ &\quad - x_{i+1,j+1,k} - x_{i+1,j,k+1} - x_{i,j+1,k+1} \\ &\quad + x_{i+1,j+1,k+1} \end{aligned}$$

The mapping to the y and z coordinates is similar.

### Electromagnetic Field Solve

The electromagnetic field portion of the code is based on the algorithm of Gedney and Lansing[1995], specialized to the case of a structured grid of hexahedral cells. This is a discrete volume generalization of the standard Finite- Difference Time-Domain (FDTD) algorithm and reduces identically to FDTD if the grid is orthogonal.

This algorithm uses a staggered grid system, as illustrated in Fig.3. For each cell (primary cell), wc define a dual cell with vertices all at half integers in the logical space. We shall call the primary cell **B** grid and dual cell **E** grid. The fundamental variables that our code calculates are **B** . ds (on each face of 13 grid cells) and E. ds (on each face of E cells). These are the quantities that have time derivatives specified on the left hand side of Eqs. 4 and they appear in the constraint equations Eqs. 5. The integrals in Eqs. 4 are taken to be over a

single face of a **B** or E cell, and in Eqs. 5 is taken to be an integral over all six faces of a **B** or **E** cell. The current J ds is co- located with **E** ds. We locate **B**·**dl** on the edges of the E grid and they pass through faces of the **B** grid. We locate **E**·**dl** on the edges of the **B** grid and they pass through the faces of the **E** grid. When the grid is Cartesian, the edges and the face normals arc colinear; and the grid system reduce to the familiar Yee lattice for Cartesian FDTD codes [Yee,1965].

We label each grid quantity with the logical coordinates of its center, e.g. a cell is labeled by the coordinates of its center and a **B** ds by the coordinates of the face center. Vertices of the **B** grid and cells of the **E** grid are integer triplets, and all other quantities have 1, 2 or 3 half integer coordinates. Faces and face variables must have a subscript 1, 2 or 3 for the logical coordinate direction of the normal (replacing the x, y, *z* subscripts in a Cartesian grid). Likewise, edges have a subscript of 1, 2, or 3 for the logical coordinate direction of the edge vector. Thus the **B**·**ds** component in direction 1 located on the face whose center is at $\mathbf{r} = (i, j + 1/2, k + 1/2)$ is labeled $\mathbf{B}\,\mathbf{ds}_{1,i,j+1/2,k+1/2}$.

To map these quantities to array locations we simply drop any half integers in the coordinates. For simplicity, below we suppress the subscript on quantities that are at location (i, *j*, k), and for quantities at nearby gridpoints we include only the coordinates that are offset. For example we write $\mathbf{E}\cdot\mathbf{dl}_{3,i,j+1,k}$ as $\mathrm{E}\ \mathbf{dl}_{3,j+1}$. With this notation, the spatially discretized curl Maxwell's equations are then:

$$
\begin{aligned}
d_t\mathbf{B}\cdot\mathbf{ds}_1 &= \mathbf{E}\cdot\mathbf{dl}_{2,k+1} - \mathbf{E}\cdot\mathbf{dl}_2 + \mathbf{E}\cdot\mathbf{dl}_3 - \mathbf{E}\cdot\mathbf{dl}_{3,j+1}\\
d_t\mathbf{B}\ \mathbf{ds}_2 &= \mathrm{E}\ \mathbf{dl}_{3,i+1} - \mathbf{E}\,\mathbf{dl}_3 + \mathrm{E}\cdot\mathbf{dl}_1 - \mathbf{E}\cdot\mathbf{dl}_{1,k+1}\\
d_t\mathbf{B}\ \mathbf{ds}_3 &= \mathbf{E}\cdot\mathbf{dl}_{1,j+1} - \mathbf{E}\cdot\mathbf{dl}_1 + \mathbf{E}\cdot\mathbf{dl}_2 - \mathbf{E}\ \mathbf{dl}_{1,i+1}\\
d_t\mathbf{E}\ \mathbf{ds}_1 &= \mathbf{B}\cdot\mathbf{dl}_{2,k-1} - \mathbf{B}\ .\ \mathrm{dl}_2 + \mathrm{H}\ .\ \mathrm{d1}_3 - \mathbf{B}\,\mathbf{dl}_{3,j-1}\\
&\quad + \mathbf{J}\ \mathbf{ds}_1\\
d_t\mathbf{E}\ \mathbf{ds}_2 &= \mathrm{B}\ \mathbf{dl}_{3,i-1} - \mathrm{B}\ .\ \mathrm{d1}_3 + \mathrm{B}\ .\ \mathbf{dl}_1 - \mathrm{B}\ \mathbf{dl}_{1,k-1}\\
&\quad + \mathrm{J}\ \mathbf{ds}_2\\
d_t\mathbf{E}\ \mathbf{ds}_3 &= \mathbf{B}\cdot\mathbf{dl}_{1,j-1} - \mathbf{B}\cdot\mathbf{dl}_1 + \mathbf{B}\cdot\mathbf{dl}_2 - \mathbf{B}\ \mathbf{dl}_{1,i-1}\\
&\quad + \mathbf{J}\ \mathbf{ds}_3
\end{aligned}
$$

By using the staggered grid, these equations will hold automatically for the discretized field divergence conditions (eqs 5)

$$
\begin{aligned}
0 &= \mathbf{B}\cdot\mathbf{ds}_1 - \mathbf{B}\cdot\mathbf{ds}_{1,i+1} + \mathbf{B}\cdot\mathbf{ds}_2 -\\
&\quad \mathbf{B}\ d\,s_{2,j+1} + \mathbf{B}\,\mathbf{ds}_3 - \mathrm{B}\ \mathbf{ds}_{3,k+1} \qquad (8)\\
\rho dV &= \mathbf{E}\cdot\mathbf{ds}_1 - \mathrm{E}\ \mathbf{ds}_{1,i-1} + \mathbf{E}\cdot\mathbf{ds}_2 -\\
&\quad \mathbf{E}\cdot\mathbf{ds}_{2,j-1} + \mathbf{E}\cdot\mathbf{ds}_3 - \mathbf{E}\cdot\mathbf{ds}_{3,k-1}
\end{aligned}
$$

Therefore, if the particle push part of the code maintain the charge conservation condition eq(3) rigorously, one can update the electromagnetic field only using eq(7) as long as the initial condition satisfies 5. The system is then time discretized by staggering **E** and **B** by *dt/2* and leapfrogging **E** and B over each other from eq(7) with time step *dt as* in the standard FDTD scheme.

Note that this set of equation is not yet complete until we specify how the edge quantities, the **E dl** and the **B** . dl, are determined from the **E** ds and the **B** ds respectively. For a uniform orthogonal grid, ds and **dl** arc parallel and we can calculate, for instance **B** . **dl** = **B** . **ds**|**dl**|/|**ds**|, making Eqs. 4 a closed system. Therefore, there is no real distinction between face and edge quantities. In this limit, the method gives second order accuracy in space. However, for nonorthogonal coordinates, **dl** is not parallel to ds, and converting face to edge quantities is more involved. To obtain **B** . **dl** at a particular cell face, one needs first to find the the values for **B** at the four vertices of the face from **B** . ds and then do a volume weighted average dotted with the dual edge vector. The technique to convert face to edge quantities is due to Gedney and Lansing[1995]. (This technique is also similar to the Discrete Surface Integral method of Madsen[1995]. ) Interested readers arc referred to Karmesin et al. [1996] and Gedney and Lansing[1995] for a detailed discussion.

Finally, once all the **B** . ds and **E** ds have been updated, wc also need to calculate the **B** and E at the vertices of the primary grid, $\mathbf{B}_{vert}$ and $\mathbf{E}_{vert}$ because $\mathbf{B}_{vert}$ and $\mathbf{E}_{vert}$ are the field vectors used to push particles. To calculate the fields at a vertex, face quantities (**B**· ds and E . ds) adjacent to this vertex arc averaged. For each vertex, the calculation of $\mathbf{B}_{vert}$ involves 12 **B** . **ds** and that of $\mathbf{E}_{vert}$ involves 8 E ds.

I'article Push

(7) The particle push part includes interpolating fields to particle Positions (gather), updating particle trajectories (particle move), and depositing particle currents (scatter). In our code, the particle position is kept in logical space while the velocity is kept in physical space. Setting the particles' fundamental shape and position in logical space simplifies both the force interpolation and the current deposit because the interpolation weights are simple linear functions of each logical coordinate. We keep the particle velocity in physical space because it is needed for use in the calculation of the Lorentz force and the current deposit.

*Gather and Scatter*

The field interpolation (gather) and c urrent deposit (scatter) are done in logical space which is Cartesian. Therefore, the numerical schemes for field interpolation

and current deposit are exactly the same as that in a flat grid electromagnetic PIC code (for instance, see Wang et al.[1995]). In particular, the current deposit is based on the charge conservation scheme discussed in Villasenor and Buneman[1992] which satisfies the discretized charge conservation condition

$$d_t \rho dV = \mathbf{J}\cdot\mathbf{ds}_1 - \mathbf{J}\cdot\mathbf{ds}_{1,i+1} + \mathbf{J}\cdot\mathbf{ds}_2 - \mathbf{J}\cdot\mathbf{ds}_{2,j+1} + \mathbf{J}\cdot\mathbf{ds}_3 - \mathbf{J}\cdot\mathbf{ds}_{3,k+1} \quad (9)$$

rigorously. Here $\rho dV$ is the total charge in a single cell of the E grid and J . ds is the current crossing a face of the **E** grid. The current deposit is done by calculating for each particle within a time step how much charge crosses each face of the **E** grid using the cells in logical space.

Particle Move

The trajectory of each particle is integrated using a time-centering leapfrog scheme for eq(1). From the velocity in physical space v, the logical space position q is readily obtained from

$$\frac{d\mathbf{r}}{dt} = R(\mathbf{r})\cdot\mathbf{v} \quad (10)$$

where

$$R(\mathbf{r}) = [\frac{\partial(x,y,z)}{\partial(r,s,t)}]^{-1}$$

is the rotation matrix. (The physical location of the particle can be easily calculated from q using the geometry information for each cell. ) The numerical scheme for particle move on non-orthogonal grid, which requires a modification to the simple leapfrog update of x and v for an orthogonal grid [Birdsall and Langdon,1985], is as follows:

1. Identify particle location at step $n$ in the logical space and interpolate the field value to the particle position $E(r^n)$, $B(m)$
2. Update v in physical space: $\mathbf{u} = \gamma\mathbf{v}$

$$\mathbf{u}^{n+1/2} - \mathbf{u}^{n-1/2} = dt[\frac{q}{m}\mathbf{E}^n + \frac{\mathbf{u}^{n+1/2} + \mathbf{u}^{n-1/2}}{2\gamma^n}\times\frac{q}{m}\gamma^n\frac{\mathbf{B}^n}{c}]$$

3. Advance r in the logical space with a predictor. corrector scheme:

3.1 calculate the rotation matrix at r", $R(\mathbf{r}^n)$

3.2 predict $\mathbf{r}^{n+1/2}$:

$$r^{"i\ 1/2} = r^{"} + R(\mathbf{r}^n)\,\mathbf{v}^{n+1/2}dt/2$$

3.3 calculate the rotation matrix at $r^{n+1/2}$ $R(\mathbf{r}^{n+1/2})$

3.4 advance **r** a full time step:

$$r^{n+1} = r^n + R(\mathbf{r}^{n+1/2})\cdot\mathbf{v}^{n+1/2}dt$$

There are two ways to calculate $R$. One way is to calculate $R$ at cacb vertex once at the start of the computation and then interpolate R to the particle position in the particle update using linear interpolation in logical space. The other is to calculate $R$ directly at the particle position when needed.

## 3. Numerical Tests and Accuracy Analysis

### Particle Push

We first test the particle push part of the code by considering single particle motions. A test case is shown in Fig.4. In this test case, we consider a domain where the $x$ coordinate is deformed from its Cartesian grid location by $\alpha\sin(my)$ and the y coordinate is deformed by $\alpha\sin(mx)$ (Fig.4a) (where $\alpha = 2$ and $m = 8\pi/L$) There is a constant background $E$ field along the y direction, $E_y$, and a constant background $B$ field along the $z$ direction, $B_z$. Two charged particle trajectories in the logical space and the physical space are shown in Figures. 4b and 4c, respectively. Fig.4c shows the correct $\vec{E}\times\vec{B}$ drift of the particles in the physical space.

In an orthogonal grid PIC code, the leapfrog scheme for particle update has second order accuracy in time and space. in the non-orthogonal grid PIG code, the particle move algorithm involves transformation between the physical space and the logical space through a rotation matrix. We next derive the accuracy of our particle move scheme.

For simplicity, let's consider a 2-dimensional domain shown in Fig.5, where the y coordinate is deformed from the Cartesian location by $\delta y = \alpha\sin mx$. Hence, the discrete grids are

$$x(i,\ j) = ih \quad (11)$$
$$y(i,\ j) = \alpha\sin(mih)) + jh$$

where $h$ is grid spacing ($h = L_x/N_x$, $L_x$ the system length, $N_x$ number of grid points). $mh$ is the grid resolution for a given distorted grid.

From eq(6), the t ransformat ion between $\Omega_p$ and $\Omega_L$ within the (i, $j$) cell is

$$x = x(i,j) + rh \quad (12)$$
$$y = y(i,j) + \alpha(\sin(m(i+1)h) - \sin(mih))r + sh$$

where $r, s \in [0, 1]$. hence the rotation matrix is

$$\frac{dx}{dt} = h\frac{dr}{dt} \quad (13)$$
$$\frac{dy}{dt} = \alpha(\sin(m(i+1)h) - \sin(mih))\frac{dr}{dt} + h\frac{ds}{dt}$$

$$= \alpha(mh\cos(mih) - m^2h^2\frac{\sin(mih)}{2} + o(m^2h^2))\frac{dr}{dt} + h\frac{ds}{dt}$$

in the limit of a continuous grid, h - › O, the grids are given by

$$x = \zeta h$$
$$y = \alpha\sin(m\zeta h) + \eta h$$

where $\zeta, \eta \in$ (–co, cm). For a continuous grid, the rotation matrix is simply

$$\frac{dx}{dt} = h\frac{d\zeta}{dt} \quad (14)$$
$$\frac{dy}{dt} = \alpha mh\cos(m\zeta h)\frac{d\zeta}{dt} + h\frac{d\eta}{dt}$$
$$= \alpha(mh\cos(mih) - m^2\delta\sin(mih) + o(m^2h\delta))\frac{d\zeta}{dt} + h\frac{d\eta}{dt}$$

where $\zeta = ih + \delta$. The difference between eq(14) and eq(13) is:

$$error = \sin(mih)\alpha m^2h(\frac{h}{2} - \delta) \sim O(\alpha m^2h^2) \quad (15)$$

This is the numerical error introduced by the rotation matrix.

For a numerical test, we consider a particle moving under no force field with an initial velocity $v_{x0}$ along the x direction in the domain shown in Fig.5. Fig.6 shows the maximum spatial error as function of $\alpha$ for three different grid resolutions $mh$ = 1,2, and4. Fig.7 shows the error in the particle's trajectory as a function of its $x$ position for different grid distortions $\alpha$, grid resolutions $mh$, and time steps $v_{x0}dt/h$. In agreement with the above derivation, we find that the error is linearly proportional to grid distortion, $\delta/h \propto O(\alpha)$, and second order in the grid resolution 6/h $\propto O(m^2h^2)$. The spatial error 6/h in the trajectory is independent of the time step $v_{x0}dt/h$. We have done numerical testing on other smoothly distorted grids, and observed same accuracy results.

Hence, for a non-orthogonal grid, the particle move scheme is still second order accurate in time and space

$$Error = O(dt^2) + O(\alpha m^2h^2) \quad (16)$$

Field Solve

We next test the electromagnetic field solve part of the code. When the grid is orthogonal, as discussed in Section 2, the field solve reduces to the standard FDTD scheme and has a second order convergence rate. When the grid is non-orthogonal, we are unable to analyze the accuracy of the field solve algorithm analytically. Hence, we perform numerical tests to study this issue.

We consider the propagation of a plane electromagnetic wave in a vacuum domain without plasma particles. We consider a rectangular box domain with length $L_x = L_y = 10$ and $L_z = 20$. The domain is initially loaded with a plane EM wave propagating along the z direction with wave number $k_z = 2\pi/L_z$. Periodic boundary condition is applied to all box surfaces.

The physical domain is discretized into a grid with $N_x, N_y$ and $N_z$ grids in each direction

$$x(i, j, k) = ih + \alpha\sin(m_xih)\sin(m_yjh)\sin(m_zkh)$$
$$y(i,j,k) = jh + \alpha\sin(m_xih)\sin(m_yjh)\sin(m_zkh)$$
$$k(i,j,k) = kh + \alpha\sin(m_xih)\sin(m_yjh)\sin(m_zkh)$$

where h = $L_l/N_l$ and ml = $2\pi/L_l$ ($l$ = $x$, y, z). A y-z cutting plane of such a domain is shown in Fig.8.

We perform numerical tests under different grid resolutions, h, and grid distortions, $\alpha$, and compare numerical solutions with the analytical one. In the numerical tests presented here, the following four sets of grid numbers ($N_x$ x $N_y$ x $N_z$) are used: 8 x 8 x 16 ($h$ = 1.25), 16 x 16 x 32 (h = 0.625), 32 x 32 x 64 ($h$ = 0.3125) and 64 x 64 x 128 (h = 0.1563). For grid distortion, we consider $\alpha$ = 0.4, 0.8, and 1.2.

In Fig.9, we fix the grid distortion at $\alpha$ = 1.2 and compare the total field energy for the four different grid resolutions as a function of time. Analytically, the total field energy is a constant at $E_{fld}$ = 1000. For the 8 x 8 x 16 grids, the error in field energy is $\delta E/E_{fld} \sim 0.02$. However, with 64 x 64 x 128 grids, there is little error in $E_{fld}$.

Since the *particle push* utilizes the field vectors at the vertices of the primary grid, $\mathbf{E}_{vert}$ and $\mathbf{B}_{vert}$, to push particles, we next evaluate the difference between the non-zero analytical field component and the numerical one at the vertices of the primary grid. We use the maximum difference between the numerical $\mathbf{E}_{vert}$ and the analytical one at the location with maximum grid distortion, Err, as a measure of the numerical error. In Fig.10 we plot $Err$ as a function of the grid resolution h for all three grid distortions as well as the orthogonal grid ($\alpha$ = O). The error shows a second order convergence ($Err \propto O(h^2)$), for the orthogonal grid, as expected. For the three non-orthogonal grids, our numerical tests show a convergence rate of $Err \propto O(h^\beta)$ with $1.6 \leq \beta \leq 1.85$.

The reduced accuracy in the *field* solve is clue to the algorithms of calculating vertex quantites ($\mathbf{B}_{vert}$ and $\mathbf{E}_{vert}$ ) and edge quantities ($\mathbf{B} \cdot \mathbf{dl}$ and E,. $\mathbf{dl}$) from face quantities ($\mathbf{B} \cdot \mathbf{ds}$ and $\mathbf{E} \cdot \mathbf{ds}$). As discussed in Section 2, a vertex quantity is obtained by averaging the face quantities adjacent to the vertex, and an edge quantity of a particular face is obtained by a averaging the face quantit ies adj scent to that face. Because of this averaging, for non-orthogonal grids the field solve gives a spatial accuracy in between the first and second order. Hence, in practical application, one should try to minimize the regions of distortion to improve the overall accuracy of the results. A direction for future work is to improve the algorithms of converting the face quantities to edge and vertex quantities.

The above results concern a grid distorted in all three directions. Next we also run the same test case for a grid distorted in two directions:

$$\begin{aligned} x(i,\ j,k) &= ih + \alpha \sin(m_x ih)\sin(m_y jh) \\ y(i,\ j,k) &= jh + \alpha \sin(m_x ih)\sin(m_y jh) \\ k(i,\ j,k) &= kh \end{aligned}$$

and a grid distorted only in one directions:

$$\begin{aligned} x(i,j,k) &= ih + \alpha \sin(m_x ih) \\ y(i,j,k) &= jh \\ k(i,\ j,k) &= kh \end{aligned}$$

In Fig. 11 we compare *Err* as a function of *h* for the 1-D, 2-D, and 3-D grid distortion case (with $\alpha$ = 0.8) and the no distortion case. For the l-I) and 2-D distortion case, the results yield a much better accuracy than the 3-D distortion case. The accuracy of the 1-D distortion case is almost the same as that of the n-distortion case. This is because in the 1-D and 2-D distortion case the direction of spatial dependence of the solution, i.e. z, is not distorted. This suggests that one may also improve the overall accuracy by minimizing the distortion of the grid in the direction of spatial dependence of the solution.

EMPIC

Finally we test the entire PIC code. For this test, we consider a counter streaming beam system: two equal electron beams arc set to counter stream in the x direccion with drifting velocities $v_d = \pm 0.4c$. The electrons within each beam follow a Maxwellian distribution with thermal velocity $v_t = 0.05c$. The ions arc considered as a fixed background. This counter streaming system generates the well-known two-stream instability

Again we consider a rectangular box domain with d istorted grids described by eq(1 7). Periodic boundary conditions are applied to all surfaces. We take grid distortion to be $\alpha$ = 0.2 and grid resolution $h$ = 1. In Fig. 12 we plot electromagnetic field energy $E_{fld}$, particle kinetic energy $E_{part}$, and total energy in the system $E_{tot}$ as a function of time. Fig.12 shows the correct behavior of energy transfer between the particles and the fields as a result of the instability excitation and saturation, while $E_{tot}$ stays as constant as it should be. In Fig.13 we compare $E_{fld}$ and $E_{part}$ for different grid distortions, $\alpha$ = O, 0.1, and 0.2. In all the three runs, the maximum fluctuation of the total energy is $\delta E_{tot}/E_{tot}$ <0.015.

## 4. Performance on a MIMD Parallel Computer

The parallel implementation of this PIC code is identical to that of our Cartesian grid based PIC code [Wang et al, 1995] The code is implemented using the General Concurrent PIC (GCPIC) algorithm [Liewer and Decyk, 1989]. The algorithm uses spatial decompositions of the physical domain to divide the computation among parallel processors. Each processor is assigned a subdomain and all the particles and grid points in it. When a particle moves from one sub domain to another, it must be passed to the appropriate processors, which requires interprocessor communication. Interprocessor communication is also necessary to exchange guard cell information for current deposit and field solve. Interested readers are referred to [Wang et al.,1995 and 1997] for detailed discussions on the parallel implementation of a EMPIC code.

In this section, we present result from running this code on the JPL/CRAT T3D parallel supercomputer. The CRAY T3D at J PL has 256 numerical nodes, each with a memory of 64 Mbytes (8 Mwords) and a peak speed of 150 Mflops. Hence, the total memory size is 16.384 Gbytes (2.048 Gwords) and the total peak speed 38.4 Gflops.

To evaluate the performance of the non-orthogonal EMPIC code, we run the code for the test case of two-stream instability described in the last section. Wc measure the total loop time per time step $T_{tot}$ as well as the times spent by each subroutine of the code for a series of runs. Since each processor runs the code with slightly different times, these times measured are the maximum processor times on a parallelcomputer.

wc perform a scaled problem size analysis, i.e. keeping the problem size on each processor fixed while increasing the number of processors used. We load the following problem: each processor has a cubic subdomain of 16 x 16 x 16 cells and 3.16 x $10^5$ particles (~ 77

particles/cell). 'l'he problem size is scaled up in three dimensions. When this problem is loaded on all 256 nodes, the total problem size is 64 x 128 x 128 (1.05 million) cells and about 80.8 million particles. The run times of this code as a function of the number of processors used, $N_p$, are shown in Fig.14. For comparison, we also plot the results from running our Cartesian grid EMPIC code on the same figure.

Fig.14a shows the total loop time, $T_{tot}$, and the time spent by the subroutines to trade particle and guard cell information between processors, $T_{comm}$. Since the global conditions are applied within the same subroutines for interprocessor communications, $T_{comm}$ is the sum of the time spent by the code on communications and global boundary conditions. When the number of processors used is much larger then one, $T_{comm}$ is dominated by the interprocessor communication time and hence it is a good measure of the parallel communicat ion cost.

We find, similar to the Cartesian grid EMPIC code, $T_{tot}$ for the non-orthogonal grid PIC code stay almost constant as the number of processors is increased indicating a high parallel efficiency. $T_{comm}$ only occupies a negligible portion of $T_{tot}$. The parallel efficiency, $\epsilon \simeq 1 - T_{comm}/T_{tot}$, is about 0.96 and 0.98 for the Cartesian and non-orthogonal grid code, respectively.

Fig. 14b shows the times spent by the two stages of the EMPIC codes. We define the particle push time, $T_{push}$, as the sum of the times for moving particles, depositing currents, applying boundary conditions, and related interprocessor communications, and the field solve time, $T_{field}$ as the sum of the times for updating the E and B fields, applying boundary conditions, and related interprocessor comniunicat ions [Wang et al., 1995]. For the Cartesian grid code, we have $T_{push} \simeq$ 7.97s and $T_{field} \simeq$ 0.039s. While for the non-orthogonal grid code, $T_{push} \simeq$ 16s and $T_{field} \simeq$ ().135s. Hence, the overall computing speed of the non-orthogonal grid code is about half of that of the Cartesian grid code. The particle push time per particle per time step $T_{push}/part$ is 98.6ns for the Cartesian grid code and 198ns for the non-orthogonal grid code. Obviously the non-orthogonal grid code involves significantly more computations (which also results in the slightly higher $\epsilon$.

## 5. Summary and Conclusions

W c h a v e developed a three dimensional non-orthogonal grid electromagnetic PI(; code for parallel supercomputers. The numerical formulation is based on hexahedral cells which are logically connected cutric cells but distorted to body-fit complex geometries in physical space. The *particle push* algorithm calculates particle velocities in the physical space but *per*forms particle tracing and grid-particle interpolations in the logical space. The *field solve* algorithm is an extension of the classical staggered mesh finite-difference time-domain (FDTD) to non-uniform meshes [Gedney and Lansing, 1995] and reduces to the standard FDTD algorithm when the grid is Cartesian. The parallel implementation of the code is almost identical to that of the Cartesian-grid EMPIC code [Wang et al.,1995] The performance of the code is evaluated on a 256-processor CRAY T3D. It is shown that the code runs with a high parallel efficiency of $\epsilon > 96\%$. However, since the non-orthogonal grid code involves significantly more calculations, the speed of the non-orthogonal grid code is only ahout a half of that of the Cartesian grid code by Wang et al[1995,1997]. Numerical experiments are performed to test the accuracy of the code. For a distorted grid, the accuracy for the particle push algorithm is second order in time and space. However, while the classical FDTD algorithm has a second order accuracy in space, the accuracy of the field solve algorithm is between first and second order for non-orthogonal grids. Therefore, the overall accuracy is restricted by that of the field solve. For grids with large distortions, the field solve algorithm may also become unstable after several thousands time steps. An outstanding area of research is to improve the accuracy of the electromagnetic field solve on distorted meshes.

## Acknowledgement

This work was supported by the AFOSR Computa tional Mathematics Program under Grant #F49620-94 1-0336. A portion of this work was carried out at the Jet I'repulsion Laboratory, California Institute of Technology, under a contract with NASA. Access to the J PL/Caltech CRAY T3D supercomputer used in this study was provided by funding from NASA Offices of Mission to Planet Earth, Aeronautics, and Space Science.

## References


[1] C. K. Birdsall and A. B. Langdon. 1985. *Plasma Physics via Computer Simulation.* McGraw-Hill, New York.

[2] S. Gedney and F. Lansing. 1995. A generalized fee-algorithm for the analysis of microwave circuit devices. submitted to *IEEE Trans. Microwave Theory and Techniques.*

[3] S. Karmesin, P. Liewer, and J. Wang. 1996 A parallel three-dimensional electromagnetic particle-in-cell code for non-orthogonal meshes. submitted to *Journal of Computational Physics.*

[4] P. C. Liewer and V. K. Decyk. 1989. A general concurrent algorithm for plasma particle-in-cell simulation codes. *Journal of Computational Physics, 85:302-322.*

[5] N. K. Madsen. 1995. Divergence preserving discrete surface integral methods for Maxwell's curl equations using non-orthogonal unstructured grid *Journal of Computational Physics, 119:34--45.*

[6] J. Villasenor and O. Buneman. 1992. Rigorous charge conservation for local electromagnetic field solvers. *Computer Physics Communications, 69: 306316.*

[7] J. Wang, P. Liewer, and V. Decyk. 1995. 3D electromagnetic plasma particle simulations on a MIMD parallel computer. *Computer Physics Communications,* 87: 35-53.

[8] J. Wang, P. Liewer, and E. Huang. 1997. 3-D electromagnetic particle-in-cell with Monte Carlo collision simulations on three MI MD parallel computer. *Journal of Supercomputing, 10: 1-18* (in press).

[9] K. S. Yet. 1966. Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media. *IEEE Transactions on Antennas and Propagations, 14: 302-307.*


## Figure Captions

Figure 1: Illustration of a cylindrical domain construtted using deformable hexahedral grids.

Figure 2: Transformation between logical space and physical space

Figure 3: Location of electromagnetic field variables on the grids.

Figure 4: Particle push test: plasma particle $\vec{E} \times \vec{B}$ drift. a) Distorted grids in physical space: x is distorted by $\alpha \sin(my)$ and y by $\alpha \sin(mx)$; b) Particle orbits in logical space; c) Particle orbits in physical space.

Figure 5: A distorted grid used in particle push test: $x(i, j) = ih$ and $y(i, j) = jh + \alpha \sin(mih)$.

Figure 6: Maximum error in particle orbit. Grid resolutions: $mh = 1$ (solid line); $mh = 2$ (dashed line); and $mh = 4$ (dotted line).

Figure 7: Local error in particle orbit. a) $\alpha = 2$, *volt/h* = 0.1; b) $\alpha = 1$, *volt/h = 0.1;* c) $\alpha = 1$, *volt/h* = 0.4. (Grid resolutions: $mh = 1$ (solid line); $mh = 2$ (dashed line); and $mh = 4$ (dotted line) ).

Figure 8: A distorted grid used in field solve test and PIC code test. The grid point at $(x, y, z)$ is distorted by $\alpha \sin(2\pi x)\sin(2\pi y)\sin(2\pi z)$.

Figure 9: Field solve test: propagation of plan electromagnetic wave. Total electromagnetic field energy. The grid distortion is $\alpha = 0.8$. The number of grid point used are 8 x 8 x 16 (dashed line); 16 x 16 x 32 (dotted line); 32 x 32 x 64 (dot-dashed line); and 64 x 64 x 128 (dot-dot-dashed line).

Figure 10: Error in vertex E field, $E_{vert}$, at location of maximum grid distortion. Grid distortions $\alpha = 0$ (solid line); $\alpha = 0.4$ (dashed line); $\alpha = 0.8$ (solid line); and $\alpha = 1.2$ (dot-dashed line).

Figure 11: Error in $E_{vert}$. No distortion (solid line); 1-D grid distortion (dashed line); 2-D grid distortion (dotted line); and 3-D grid distortion (dot dashed line). For all distorted grids, $\alpha = 0.8$.

Figure 12: PIC code test: two-stream instability. Total energy, field energy, and particle energy. Grid distortion $\alpha = 0.2$. Grid resolution $h = 1$.

Figure 13: Comparison of the particle energy and field energy for no grid distortion (solid line), distortion with $\alpha = 0.1$ (dashed line), and distortion with $\alpha = 0.2$ (dashed line).

Figure 14: Run times of a Cartesian grid EMPIC code (solid line) and the non-orthogonal grid EMPIC code

(dashed line) on a 256-processor Cray 131). a) Total loop time $T_{tot}$ and communication/boundary condition time $T_{comm}$. b) Particle push time $T_{push}$ and field solve time $T_{field}$.

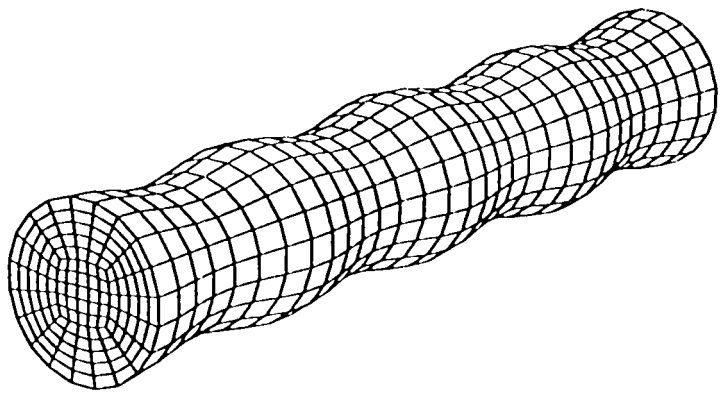


Figure 1

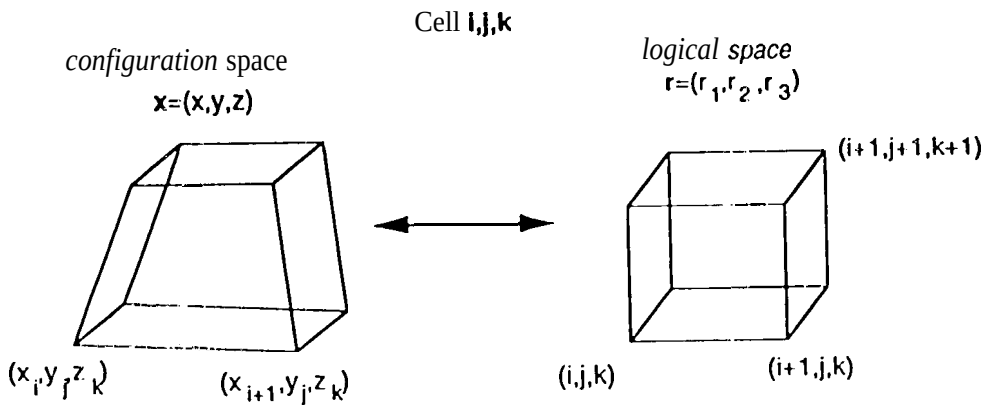



Figure 2

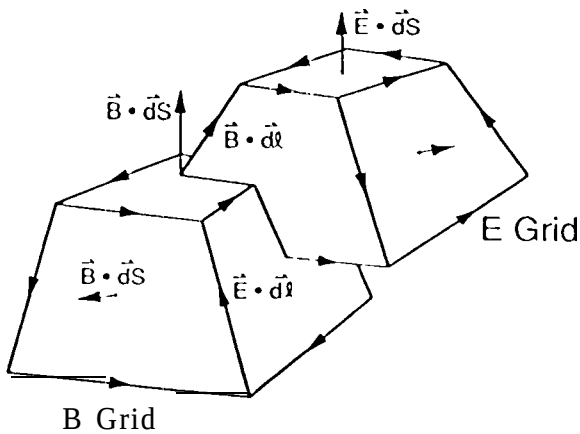
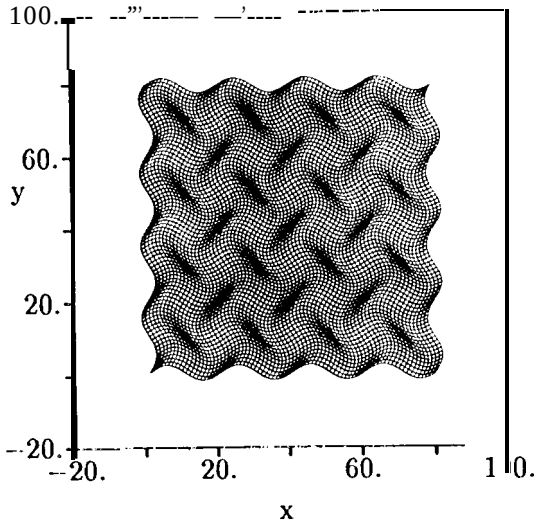
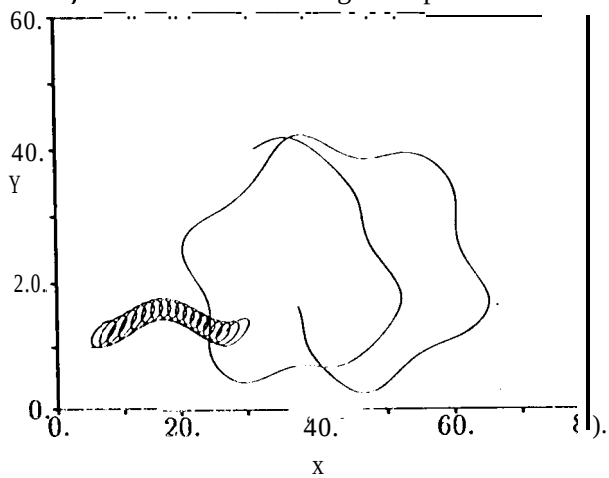



Figure 3

a) Distorted Grids in Physical Space




b) Particle Orbit in Logical Space




c) Particle Orbit in Physical Space

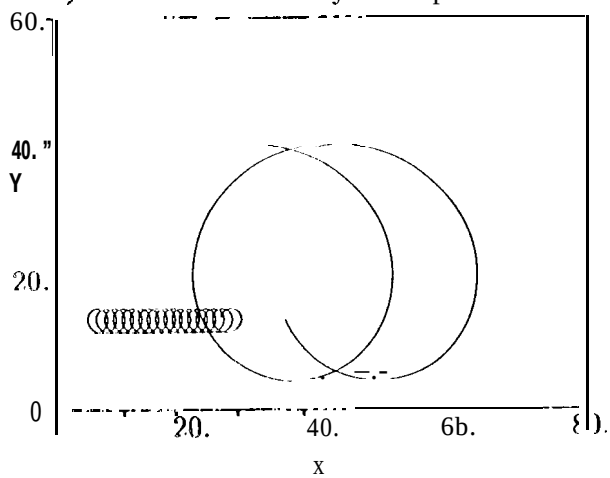



Figure 4

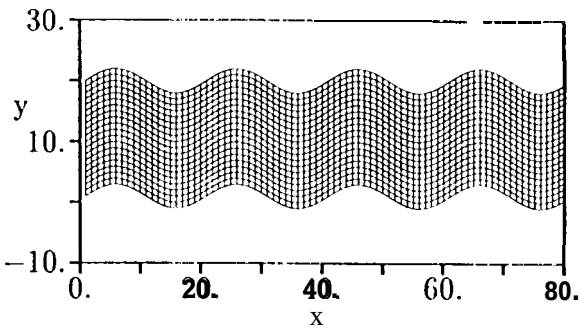



Figure 5

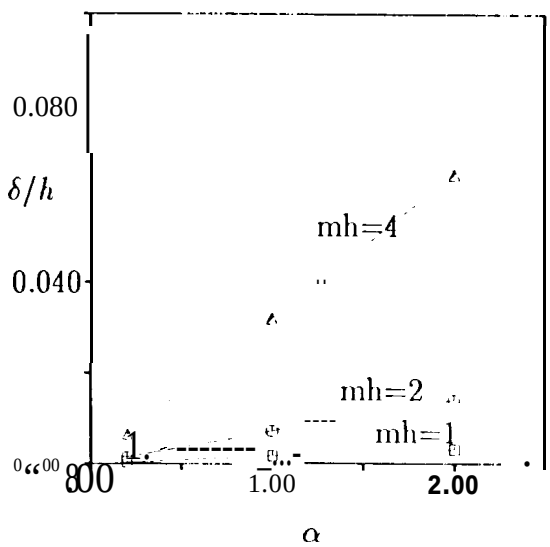



Figure 6

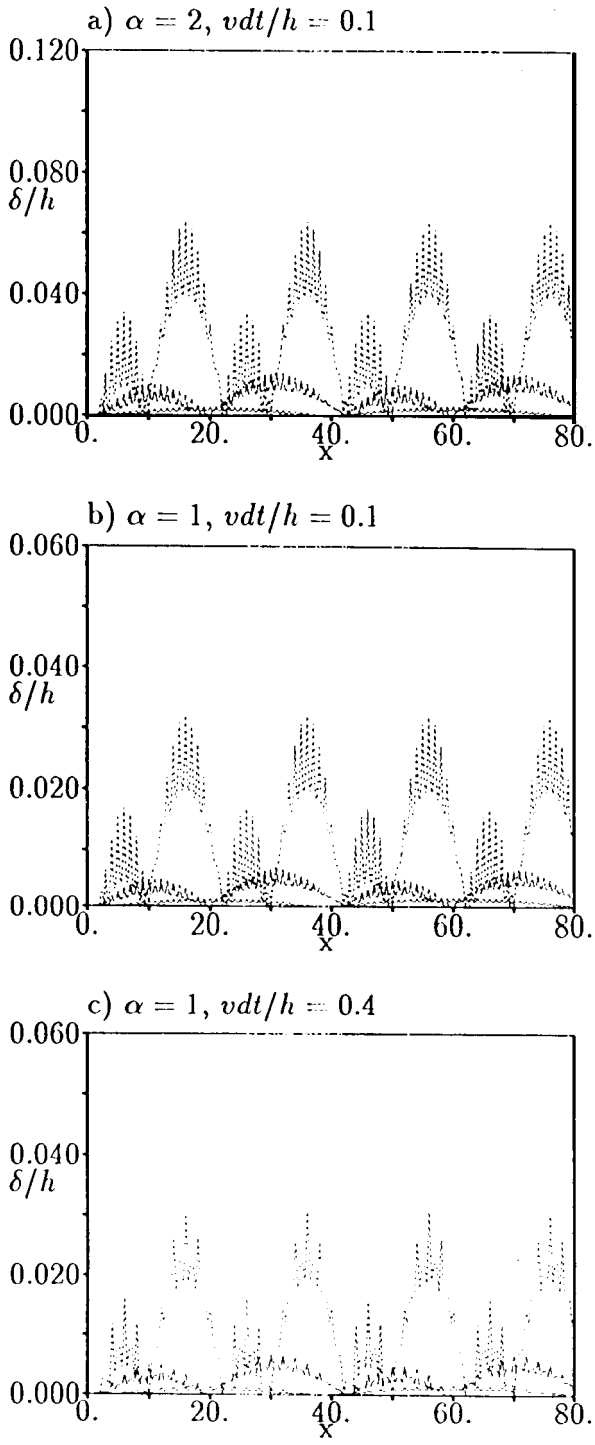



Figure 7

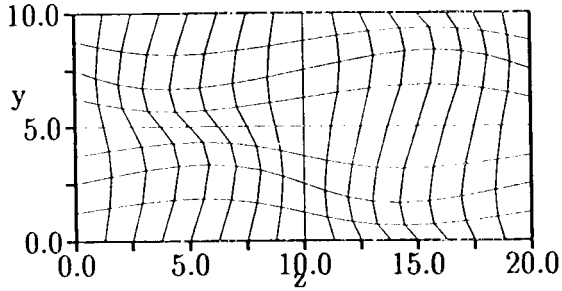



Figure 8

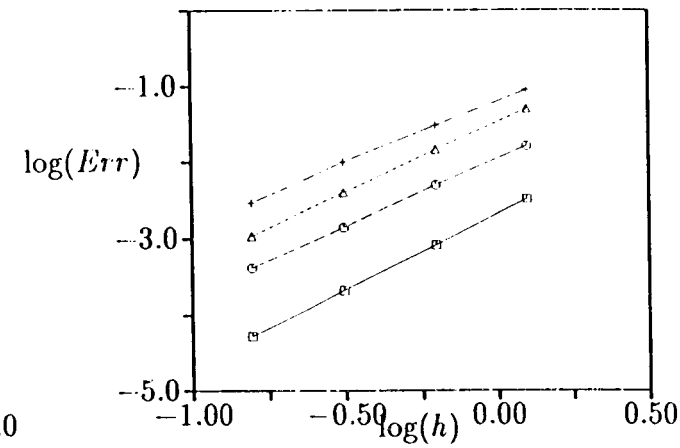



Figure 10

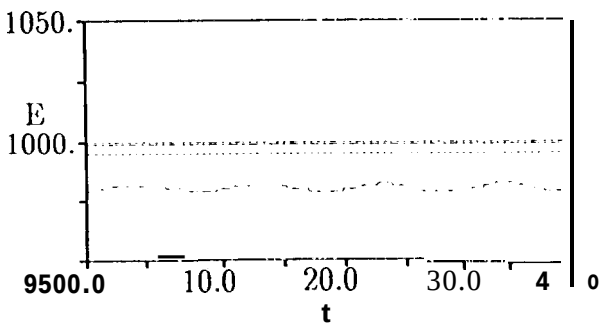



Figure 9

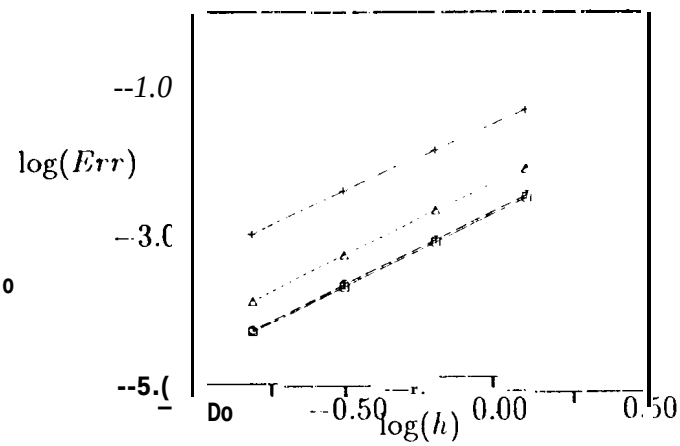



Figure 11

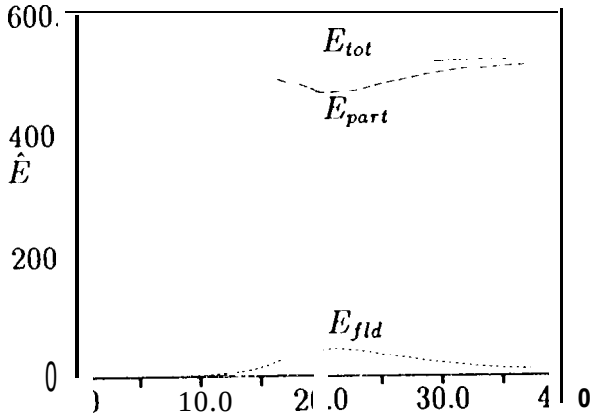
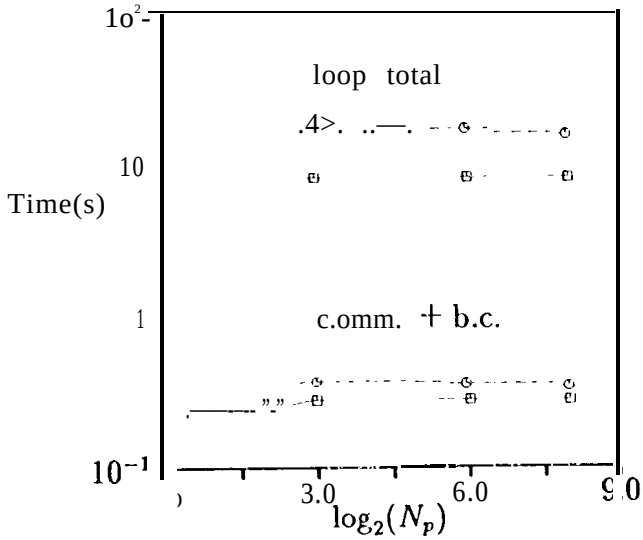



Figure 2

a) $T_{tot}$ and $T_{comm}$




b) $T_{push}$ and $T_{field}$

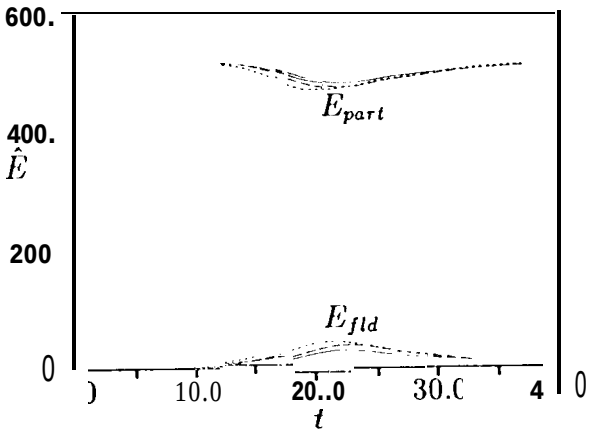



Figure 13

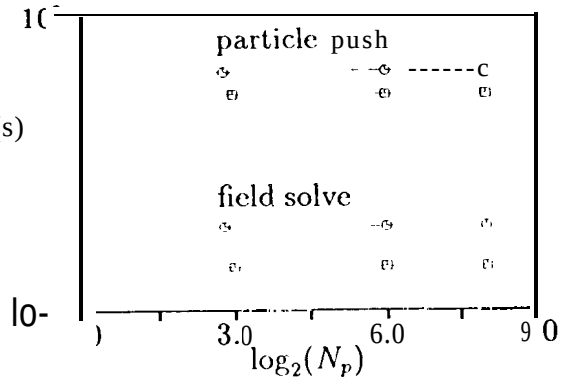



Figure 14